NEURAL NETS FOR MESH ASSESSMENT

Saeed Iqbal [1] and Graham .F. Carey [2]

[1]Department of Electrical Engineering, University of Texas at Austin, Austin, Texas, U.S.A., iqbal@ece.utexas.edu,
[2]TICAM, University of Texas at Austin, Austin, Texas, U.S.A., carey@cfdlab.ae.utexas.edu.

ABSTRACT

We investigate the construction, training and application of a neural net for assessing element shape quality of practical unstructured grids arising in mesh generation, adaptive refinement and moving grid applications. Results of numerical experiments are included to validate the process and demonstrate performance of the neural net for both triangulations in 2D and tetrahedral tessellation in 3D.
Keywords: Adaptive mesh refinement, Finite element shape quality, Neural nets.

1. INTRODUCTION

Mesh generation techniques on practical domains will lead to cells of varying shape and size. It is also desirable to grade or adaptively refine and coarsen the mesh in finite element and finite volume simulation [14]. An adaptive mesh is particularly important in three-dimensional boundary value problems discretized by finite element or finite volume methods, because the problem size and computational cost grow very rapidly under uniform refinement [13,14]. However, the accuracy and computational reliability and efficiency may be compromised if the elements in the unstructured grid are of "poor geometric quality". Moreover, in explicit time integration of transient problems, poor element shape can seriously reduce time-step size [16], which in-turn increases the CPU time to complete a simulation. Finally, in Lagrangian calculations and similar formulations that involve changing geometry and moving meshes the local element deformation can limit the range of viable simulations.

Hence, a key question in mesh generation, mesh refinement and moving mesh schemes is *assessment* of the shape quality of elements. This is a topic of increasing concern in large scale simulations with automatic grid generation and refinement algorithms and even more so when dealing with deforming moving grids. Several metrics have been proposed to be used as indicators of cell shape quality [16] but it is clear that any single indicator can provide only a very limited perspective. One can, of course, combine several such indicators, to obtain a more robust approach and use ideas from multi-objective optimization with weights specified by the analyst for a given class of problems. Another novel approach would be to train a neural net using a number of metrics for cell quality and have the net then assess the cell quality level in practical grids. In the present work we explore the use of neural nets in this context.

We emphasize, however, that assessment of the mesh shape quality is but one example where neural nets might be put to acceptable use in evaluating a system or structure. A similar notion could be applied to the error indicators commonly used to guide mesh refinement indicators for model reliability, indicators for structural damage, pattern recognition and similar "quality measures". For example the neural net could be trained to refine or coarsen a grid based on a number of a posteriori error indicators. *However*, as a cautionary note we mention that the neural net would be less advisable in applications where failure of the net to produce a precise discrimination could lead to catastrophic breakdown of the subsequent simulation. Such a situation could arise in the boundary value problem setting if a few poor elements were admitted by the net and this led to an ill conditioned system or punitive step-size restriction. Other applications such as interpolation on irregular grids typically are less sensitive and the net can be applied more confidently. Clearly, there are many other problems involving decision making where training a net on computationally intensive tasks and then using the neural net for system assessment will be even more effective.

In the next section we briefly summarize some of the key steps in the evolution of neural nets and then proceed to describe their application to mesh quality assessment.

2. HISTORICAL BACKGROUND

In 1943 McCulloch and Pitts [1] suggested a way to model events in the nervous system, and showed that a large number of very simple elements can be combined to make a network that can in principle, compute any computable function. This influenced work by Von Neumann to idealize some components mentioned in the paper and use them in the design of the EDVAC (Electronic Discrete Variable Automatic Computer) [30]. Later, in "The Organization of Behavior" Hebb [2] suggested a learning rule for synaptic modification. This rule is now the basis of many learning algorithms and has had a profound effect on the way machine learning systems were designed. In particular, neural nets were studied in the context of learning, adaptive systems, stability, and information theory [2]. For example, Rochester et. al. [3] used computer simulation to test Hebb's rule and there were several subsequent attempts by others to simulate

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **2005**	2. REPORT TYPE	3. DATES COVERED **-**
4. TITLE AND SUBTITLE **Neural Nets for Mesh Assessment**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **DOD High Performance Computing Modernization Program Ofc,Programming Environment & Training (PET),1010 North Glebee Road Suite 510,Arlington,VA,22201**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR’S ACRONYM(S)
		11. SPONSOR/MONITOR’S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT
We investigate the construction, training and application of a neural net for assessing element shape quality of practical unstructured grids arising in mesh generation, adaptive refinement and moving grid applications. Results of numerical experiments are included to validate the process and demonstrate performance of the neural net for both triangulations in 2D and tetrahedral tessellation in 3D.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**9**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

neural nets. Minsky's classic paper, "Steps Toward Artificial Intelligence" [4] introduced neural nets, similar to those used today. Following this, Rosenblatt[5] introduced a new approach to pattern recognition called the perceptron and showed an important convergence property. This motivated further activity among the research community.

For the first time trainable networks were being exploited, and it was thought that neural nets were universally applicable for problem solving. However, in 1969, Minsky and Papert [6] formally showed a fundamental limitation of single layer perceptrons. Later Hopfield networks [7] used the idea of energy functions to explain new ways of understanding computation. Several key developments occurred in the theory and understanding of learning systems: Kohonen [8] presented the idea of self-organizing maps; Ackley, Hinton and Sejnowski [9] used simulated annealing to develop a learning machine and showed that it was possible to build complex learning machines; Rumelhart, Hinton, and Williams [10], reported a computationally efficient learning algorithm for neural networks, called back propagation. This back propagation strategy is the most popular learning algorithm in use today. Mead[29], described a number of concepts combining ideas from neurobiology and VLSI technology.

Linsker [22] suggested the *Infomax* principle which applies information theory to model neural net behavior. This work was further extended by Bell and Sejnowski [23] . They and others, used information theoretic models for solving a broad class of problems [22]. For example, Broomhead and Lowe [24] proposed a new way to design multi-layered feedforward networks using radial basis functions(RBF) as an alternative to multilayered perceptrons. Subsequently, Vapnik [25, 26] designed a new kind of learning network called *support vector machines* and suggested using the so-called *Vapnik-Chervonenkis* (VC) dimension of a training data set to estimate the capacity of a neural net to learn efficiently from the data. Freeman [27] studied neuron activity and used chaos theory to describe emergence of self-organizing activity patterns in populations of neurons [12]. The accelerated advances in computer resources and increasing complexity of applications are motivating a renewed interest in neural nets. In the present work we explore the use of multilayered perceptron neural nets for mesh quality assessment.

3. NEURAL NET ARCHITECTURE

Neural nets or multi-layered perceptrons are connected layers of neurons and may be "trained" to learn specific concepts from examples. Basically, there are four key parameters that characterize a neural net architecture: (1) the number of layers, (2) the number of neurons in each layer, (3) the kind of connectivity among layers and (4) the kind of activation function used within each neuron.

A neural net can in principle have any number of layers with each layer having a number of neurons. The most common neural net architecture is comprised of 3 layers: The first layer called the input layer, a second layer called the hidden layer and the third layer called the output layer. In figure 1, we show a simple 3-layered neural net. A neural net has *k-l-m* architecture if it has *k* neurons in the first, *l* neurons in the second and *m* neurons in the third layer.

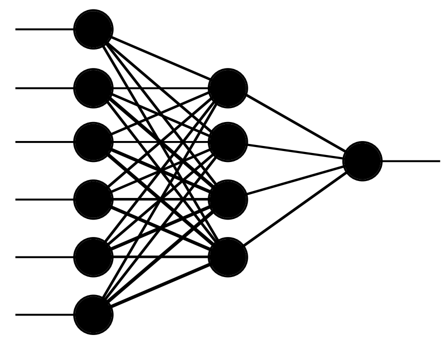

Figure 1: A multi-layered perceptron: Three vertical layers of neurons.

Usually, the number of neurons in the first layer is equal to the number of features (or inputs) available in data. The number of neurons in the output layer is at least one. In the present work the output neuron corresponds to a single quality metric. The number of neurons in the middle layer needs to be adjusted during training. Usually, there is a trade-off between accuracy of classification and number of neurons in the middle layer. Complex classification tasks require a large number of middle layer neurons. Connectivity among layers can be full or partial. If each neuron in a layer is connected to each neuron in the next layer it is called a fully connected architecture. We use a fully connected architecture in our study.

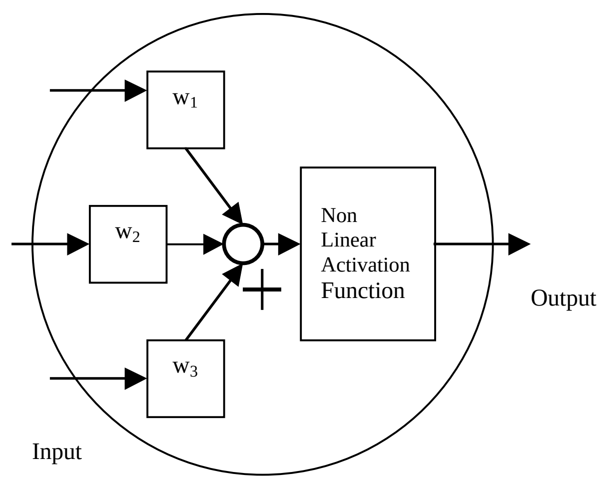

Figure 2: The functionality of a single neuron is represented by a circular region: A weighted sum of the inputs to the neuron is calculated and this sum is used as input to a nonlinear activation function which generates the output.

The behavior of each neuron is controlled by an activation function, which controls the output of the particular neuron. The nonlinear nature of the activation function is the key to the neural net capabilities. The most common activation function is the sigmoid:

$$y(x) = \frac{1}{1 + \exp(-x)}$$

where y is the output of the neuron, x is the weighted sum of all its inputs. (The weights are determined during training. The sigmoid has a continuous derivative and gives closed form expressions for weights calculation during training [12]). The functionality of a single neuron is shown in Figure 2. A weighted sum of inputs from the neurons in the previous vertical layer is computed and transformed by a non-linear activation function to an output. The weights w_i in Figure 2 are determined by the training process.

During the training process we divide all available data related to a problem into two equal sets, namely a "training set" and "test set", by random selection. Note that a portion (20-40%) of the training set is typically used for checking convergence of the neural net weights during training. We refer to this portion of the training set, as the "convergence set".

Before a neural net is useful it has to go through a training process. Training is the adjustments of weights to learn to classify accurately. Neural nets are trained by incrementally modifying the weights of connections between neurons according to a learning algorithm. Neural nets are trained by repeated exposure to training set patterns. Training is a computationally expensive process that requires some judgment for selection of training parameters and neural net architecture. The most popular training algorithm is called *back propagation* [1,2] and proceeds as follows:

1. *Initially the weights are assigned at random.*
2. *A number of training examples are "shown" to the neural net.*
3. *The output of the network is compared to the desired output and the difference between the actual output and the desired output is the error.*
4. *The total mean square error for the entire training example is calculated and the error is propagated "backwards" i.e. from the output layer to the input layer. Along the way, connection weights are adjusted to reduce the total mean square error.*

The computational cost of training grows nonlinearly with the number of layers and number of neurons (assuming the neural network is fully connected).

The aim of training is to minimize the total mean square error of the training set without over-fitting. The total mean square error is defined as follows:

$$Error_{RMS} = \frac{1}{N} \sum_{n=1}^{N} \sum_{i=1}^{O} e^2(n,i)$$

where N is the number of patterns, O is the number of neurons in the output layer, and $e(n,i)$ is the error of neuron of the output layer when the n^{th} pattern is applied to the input layer. The error at a particular output neuron is defined as the difference between the output value of the neuron and the desired value (supplied with the training set pattern). Usually, during training the value of the total mean square error of the training set and convergence set decreases. The decrease is rapid during the initial training, but converges to a value after a number of iterations. Over-fitting occurs when the total mean square error of the convergence set starts to increase. The training of a neural net is stopped when the total mean square error of the convergence set is a minimum. The corresponding total mean square error of the training set determines the accuracy of the neural net. If the accuracy of the neural net is not acceptable we have to modify its architecture and/or learning algorithm parameters to achieve higher accuracy.

Assuming that a neural net achieved has the desired accuracy, it can be used to classify unseen patterns. The features of the unseen pattern are applied to the input layer. Inputs to each neuron in the first layer are transformed by an activation function and transmitted to the neurons in the next layer and so on. The output from this trained neural net is the assessment of the input based on the training. In this study, neural nets are trained as 2-class classifiers, i.e., when the input pattern corresponding to a particular finite element is "shown" to the trained neural net it will classify it into one of the two classes (acceptable or unacceptable). The performance of the classifier is judged by considering the number of true positive, true negative, false positive and false negative classification cases. True positive are elements of acceptable quality classified as "acceptable"; similarly true negative are elements of unacceptable quality classified as "unacceptable". False positive are elements of unacceptable quality misclassified as "acceptable ", and false negatives are elements of acceptable quality misclassified as "unacceptable". An ideal classifier should have a zero false positive and false negative count.

4. APPLICATION TO ELEMENT QUALITY

Our treatment and numerical experiments here are confined to shape quality of the triangle and tetrahedron but the extension to other criteria and types of cells is immediate. There are several measures of element quality (degeneracy) suggested for simplices in the literature: See, for instance, Carey and Plaza[13], Whitehead[15], Stynes[17], Levin[18], Conti et. al. [19], Knupp [28] and Liu and Joe [20,21]. Most of these criteria are based explicitly on geometric concepts as one might expect. We have approached the problem of qualitative assessment from the machine learning point of view. Given a metric such as the ratio of the inscribed sphere radius to the circumsphere radius as a basis for determining "acceptable elements", how can we train and apply a neural net to judge the quality of any given mesh ? More importantly, since there are a number of popular metrics that are individually in use, can a neural net be applied competitively to assess mesh quality based on training with these several metrics? This latter approach affords a broader assessment than a single metric and would be closer in some sense to the assessment an analyst would make by inspection.

4.1 A SIMPLE EXAMPLE

Let us first use the simple ratio metric to illustrate the training process and then subsequently carry out more extensive training and tests with a combination of several metrics. Consider any nondegenerate triangle and let *R* be the radius of its circumcircle. For convenience, translate the circle so that its center is at the origin . Next, scale the coordinates so that the transformed circle has unit radius (Figure 3). The ratio metric (or any other metric) can be applied to the resulting triangle for a specified quality tolerance as acceptable or to be rejected.

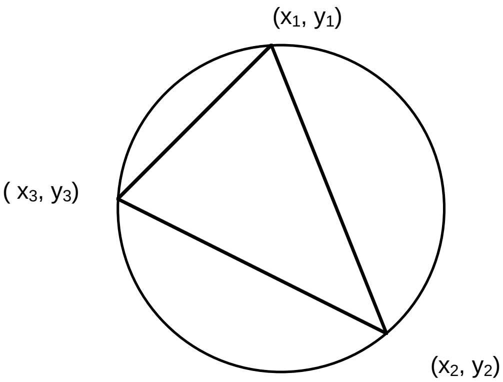

Figure 3: A triangle scaled and translated in a unit circle. The coordinates of the triangle are used to calculate input features to the neural net.

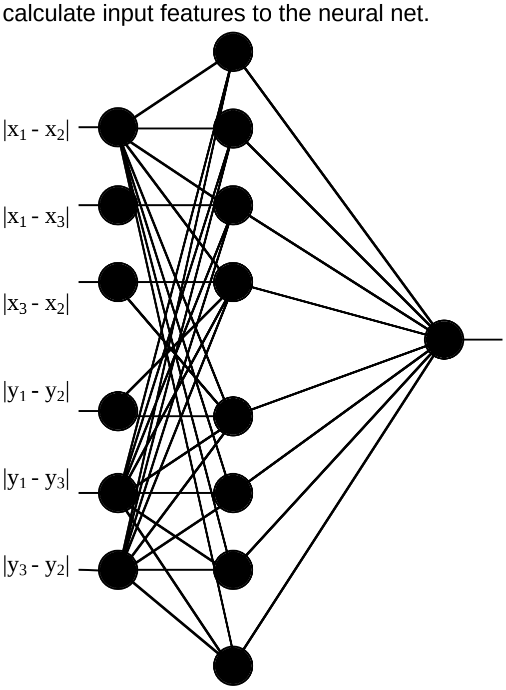

Figure 4: A 6-8-1 fully connected neural net is used to judge the quality of a triangle. Transformed coordinates are used as input.

We generate 5000 triangles by connecting random triples of points on a unit circle, and assess the quality of triangles by calculating the ratio of inscribed to circumcircle circle. All triangles with the ratio of inscribed circle to the circumcircle

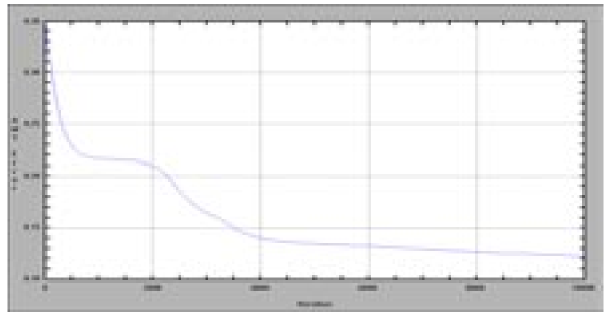

Figure 5: Decrease in the RMS error training during the training.

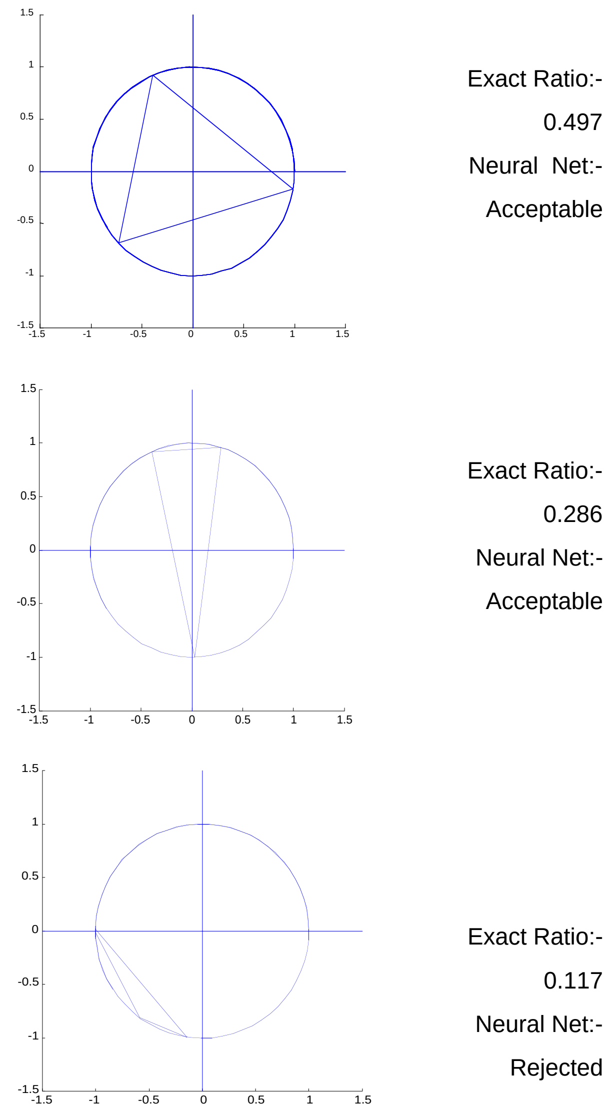

Figure 6: The output of the neural net on some examples from the data set.

less than 0.15 are classified as “rejected” else it is classified as “acceptable ”. This data is then divided into two equal sets of 2500 each. The first set is called the “training set” and is used for training various neural nets. The second is the “test set”. After training, a 3 layer 6-8-1 network shown in Figure 4 is selected. The RMS error during training using back propagation is illustrated by the Figure 5. The decrease in the RMS error rate is not uniform because parameters of the learning algorithm are interactively changed during the training. In this case the neural net achieved a high accuracy after 10000 iterations. Finally, after adequate training the net can be used to assess elements in the test set.

Results after assessing the test set show that the trained neural net can assess 98% of the triangles of the test set “correctly”. Three triangles from the test set are shown in Figure 6, along with the exact ratio metric and classification by the neural net.

4.2 A MULTIPLE METRIC 2D EXAMPLE

Consider again a set of 10000 triangles generated by connecting random triples of points on a unit circle. Next, we select following four simple quality criteria:

1. Minimum side length $S_{min} > 0.55$
2. Minimum angle $\theta_{min} > 0.4$ radians
3. Maximum angle $\theta_{max} < 2.7$ radians
4. Area $A_{min} > \pi/15$

These values correspond to the average of the entire data set. To qualify as a acceptable triangle an element must now meet all the above criteria. Based on the combined four criteria 3463 triangles were classified as acceptable and 6537 triangles failed to satisfy one or more of the criteria mentioned above, hence were rejected.

The data set divided into two equal sets namely, training set and test set. The combined data now can be viewed as a standard 2-class problem with 6 input features.

Several neural nets were trained, on the 5000 triangles training set data, having different numbers of hidden layer neurons. The architecture of 6 input, 8 hidden and 4 output nodes was selected, because it offered a combination of high accuracy and low neuron count. The performance of the classifier is shown in Tables 1 and 2. Figure 7 below shows the RMS error of the training set. Table 1 shows an overall accuracy of above 97%. Table 2 shows the details of the classifier performance.

Table 1: Combined four criterion on the randomly generated 2D data set

Combined performance on three criteria					
Elements	Metrics		Neural Net		% Correct
	Accept	Reject	Accept	Reject	
5000	2618	2382	2614	2386	97.84

Table 2: Performance of Neural net as a 2-class classifier.

2-Class Classification Performance				
Elements	True		False	
	Positive	Negative	Positive	Negative
5000	2321	2553	61	65

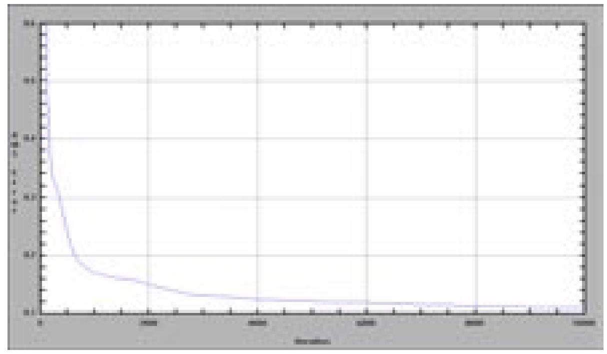

Figure 7: RMS error of the training data set 2D case.

4.3 MULTIPLE METRICS 3D EXAMPLE

To train a neural net for tetrahedron quality judgement, 10000 random tetrahedra were simailarly generated, by repeatedly selecting four points at random on a unit sphere. The following three quality criteria where selected:

1. Minimum side length $S_{min} < 0.45$
2. Distance of centroid to any surface D < 0.075
3. Volume of element $V_{min} < 0.05$

These 10000 tetrahedra are divided into equal groups of training and test data sets. After training several neural nets, the neural net architecture with 18 input units, 9 middle layer units and 3 output units corresponding to the three quality criteria, is selected. Convergence took 30000 iterations as shown Figure 8 below. The performance of the neural net is shown in Table 3 and 4. Table 3 shows an overall accuracy of above 85%. The classifier accuracy is less compared to the previous case because (1) assessment of a 3D element is more complex. (2) randomly generating tetrahedron generate a wide range of elements. Table 4 shows the details classifier performance

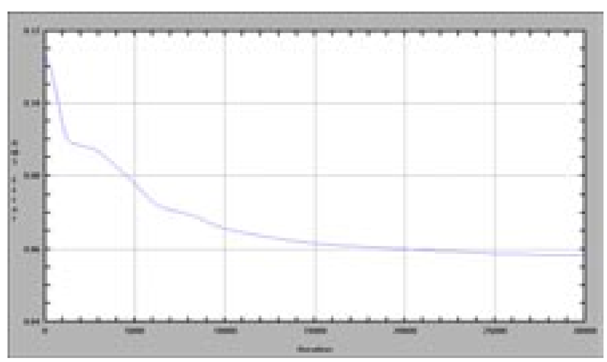

Figure 8: RMS error of the training set 3D case.

Table 3: Performance of Neural net as a 2-class classifier.

Combined performance on three criteria					
Elements	Metrics		Neural Net		% Correct
	Accept	Reject	Accept	Reject	
5000	1403	3597	1833	3167	85.88

Table 4: Performance of Neural net as a 2-class classifier.

2-Class Classification Performance				
Elements	True		False	
	Positive	Negative	Positive	Negative
5000	3029	1265	568	138

4.4 ASSESSING A 2D DEFORMING MESH

The Taylor Anvil problem is a standard benchmark problem in impact mechanics: a solid cylinder impacts a rigid flat plate normally and at high velocity. The cylinder deforms and shock waves propagate through the cylinder. Finite deformation and plastic flow are significant. In the present case we consider results from a Lagrangian formulation of the problem.

Here the mesh deforms as the shape evolves during impact. Elements in the mesh deform and the timestep of the calculation is reduced as element quality degenerates. Moreover in our Lagrangian analysis algorithm the mesh can be modified adaptively based on local error analysis and shape quality. Adaptivity is based on a point insertion and element removal algorithm. Since the problem is symmetric it can be solved for a symmetric 2D mesh of triangle elements. The initial domain is discretised as a uniform triangulation of 891 elements for the symmetric cylinder.

In this study we take the element data corresponding to the evolving Lagrangian grid at 30 different times (stages) during the impact simulation history, and combine them to make a large data set. Next we sample the large data set and randomly selected 2700 triangles for training. We apply the following four criteria to assess element quality:

1. Minimum Side Length < 0.80
2. Minimum Angle < 0.45 radians
3. Maximum Angle > 2.25 radians
4. Area of Triangle < 0.30

A 6-10-4 architecture showed best accuracy and low neuron count. This trained neural net is used to assess element quality on each of the 30 stages. The results are shown in Tables 5 and 6. A pictorial view of the assessment the mesh is shown for stage 30 is shown in Figure 9. Here the elements in the figure are color mapped according to the combined quality criteria.

4.5 ASSESSING A 3D ADAPTIVELY REFINED DOMAIN

In this case we consider an adaptive mesh refinement example from [13]. The geometry is fixed as an L shaped domain, but the tetrahedral mesh adapts with the transient solution on the domain. The adapted mesh at 14 distinct time stages was taken for input data sets and tested with a neural net.

Table 5: Combined four criterion on the Taylor Anvil problem.

	Combined performance on four criteria					
Stage	Elements	Metrics		Neural Net		% Correct
		Accept	Reject	Accept	Reject	
1	891	879	12	879	12	99.78
2	891	879	12	879	12	99.78
3	891	879	12	881	10	99.33
4	1012	994	18	994	18	98.42
5	1011	987	24	987	24	98.62
6	1011	986	25	989	22	98.52
7	1315	1286	29	1288	27	98.78
8	1423	1391	32	1381	42	97.89
9	1423	1381	42	1369	54	97.33
10	1506	1452	54	1436	70	97.74
11	1506	1438	68	1426	80	98.14
12	1543	1467	76	1456	87	98.12
13	1543	1456	87	1449	94	97.73
14	1540	1441	99	1439	101	97.14
15	1540	1438	102	1430	110	96.62
16	1611	1490	121	1490	121	96.52
17	1699	1559	140	1561	138	96.82
18	1698	1580	118	1581	117	96.88
19	1698	1439	259	1465	233	96.58
20	1698	1301	397	1332	366	93.93
21	1694	1446	248	1478	216	93.27
22	1694	1274	420	1338	356	91.74
23	1691	1314	377	1398	293	90.42
24	1803	1354	449	1441	362	91.18
25	1799	1401	398	1490	309	91.05
26	2013	1502	511	1580	433	91.75
27	2134	1494	640	1596	538	91.85
28	2134	1631	503	1762	372	90.86
29	2134	1531	603	1665	469	91.28
30	2142	1456	686	1599	543	90.15

The training set is made of 3500 randomly selected tetrahedra from all 14 stages. These are translated and scaled in a unit sphere, and assessed for quality using the following 3 criteria:

1. Minimum side length S_{min}< 0.75
2. Distance of centroid to any surface D < 0.1
3. Volume of element V_{min}< 0.175

An element in the domain is considered unacceptable if any of the above is true. We now carry out a multi-metric training procedure with simple metrics and specified several tolerances. We emphasize that these metrics are not chosen with the idea that they are best in any sense. On the contrary, the point of the exercise is to demonstrate by example. Hence, we have deliberately chosen simple primitives for the training. An 18-9-3 architecture neural net is used to assess the quality of elements in all 14 stages. Performance results are shown in Tables 7 and 8.

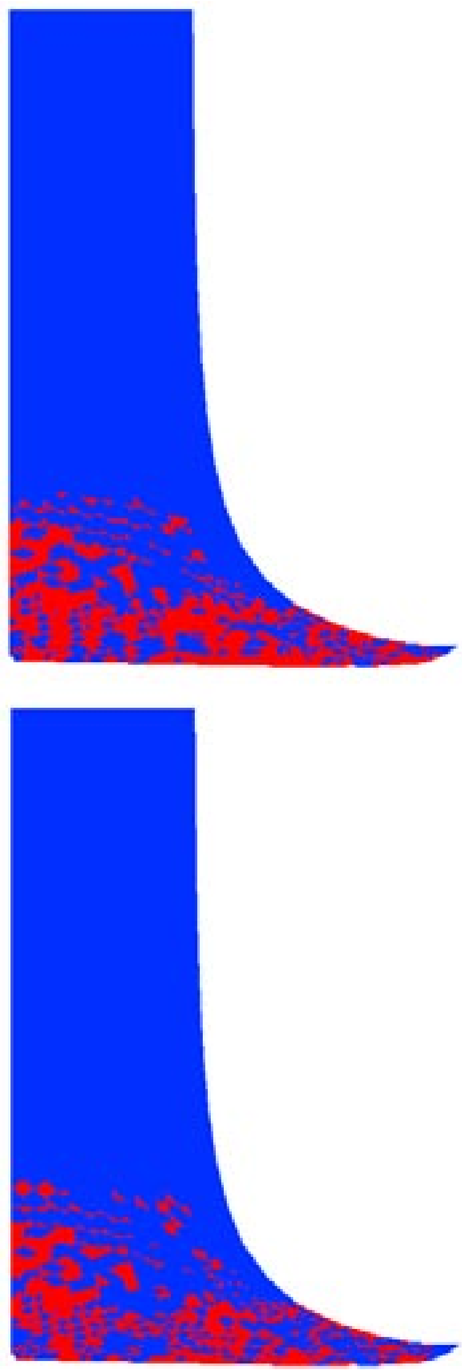

Figure 9: Comparison of element quality predicted by the neural net and exact calculation at stage 30. The figure on the top is based on exact calculation and the bottom figure shows the shape quality as it is predicted by the neural net.

Table 6: Performance of Neural net as a 2-class classifier.

	2-Class Classification Performance				
Stage	Total	True		False	
		Positive	Negative	Positive	Negative
1	891	878	11	1	1
2	891	878	11	1	1
3	891	877	8	4	2
4	1012	986	10	8	8
5	1011	980	17	7	7
6	1011	980	16	9	6
7	1315	1279	20	9	7
8	1423	1371	22	10	20
9	1423	1356	29	13	25
10	1506	1427	45	9	25
11	1506	1418	60	8	20
12	1543	1447	67	9	20
13	1543	1435	73	14	21
14	1540	1418	78	21	23
15	1540	1408	80	22	30
16	1611	1462	93	28	28
17	1699	1533	112	28	26
18	1698	1554	91	27	26
19	1698	1423	217	42	16
20	1698	1265	330	67	36
21	1694	1405	175	73	41
22	1694	1236	318	102	38
23	1691	1275	254	123	39
24	1803	1318	326	123	36
25	1799	1365	273	125	36
26	2013	1458	389	122	44
27	2134	1458	502	138	36
28	2134	1599	340	163	32
29	2134	1505	443	160	26
30	2142	1422	509	177	34

Table 7: Combined three criteria on the L-shaped domain.

	Combined performance on three criteria					
Stage	Elements	Metrics		Neural Net		% Correct
		Accept	Reject	Accept	Reject	
1	128	122	6	122	6	100.00
2	3238	3094	144	3085	153	99.16
3	1680	1653	27	1650	30	99.82
4	3298	3153	145	3157	141	99.75
5	1905	1850	55	1848	57	99.89
6	6104	5386	718	5378	726	97.74
7	2547	2391	156	2360	187	98.62
8	4811	4369	442	4352	459	98.98
9	1088	1070	18	1070	18	100.00
10	2434	2281	153	2270	164	98.89
11	1439	1366	73	1370	69	99.40
12	4145	3519	626	3509	636	95.94
13	2171	1900	271	1864	307	97.23
14	3017	2661	356	2624	393	97.71

Table 8: Performance of Neural net as a 2-class classifier.

Stage	Total	2-Class Classification Performance			
		True		False	
		Positive	Negative	Positive	Negative
1	128	122	6	0	0
2	3238	3076	135	9	18
3	1680	1650	27	0	3
4	3298	3151	139	6	2
5	1905	1848	55	0	2
6	6104	5313	653	65	73
7	2547	2358	154	2	33
8	4811	4336	426	16	33
9	1088	1070	18	0	0
10	2434	2262	145	8	19
11	1439	1364	67	6	2
12	4145	3430	547	79	89
13	2171	1852	259	12	48
14	3017	2608	340	16	53

5.CONCLUSIONS

In this study we have investigated some aspects of training and application of neural nets to mesh quality assessment. As noted in the introduction this should be prefaced by the cautionary remark that the net may not detect all poor elements and therefore this approach will be of limited value in certain applications. Further, for the simple metrics considered here the net does not provide any significant saving in CPU time on current serial processors. However, it is competitive on the simple multi-metric cases considered in the numerical experiments. The value of the neural net may lie in other applications where the cost of the metric evaluation is more expensive and where the cost of training can be more easily amortized over many assessments. Nevertheless, the approach does look interesting in several respects even in the simple mesh quality context. For instance, training can be with respect to a number of different metrics which will lead to more robust meshes and this has been an issue of interest to the meshing community where a variety of shape quality metrics have been proposed. Furthermore, other criteria can be introduced related to mesh smoothing as seen in the experiments and one can possibly apply the ideas in guiding mesh refinement based on a number of error/feature indicators. Finally, the net may be a tool for evaluating error indicators in adaptive strategies.

ACKNOWLEDGEMENTS

We express our thanks to J. P. Suárez and A. Plaza for providing the mesh for the L shaped domain and to David Littlefield for the 2D Taylor anvil mesh sequence. This research has been supported by ASCI Contract #B347883, DoD PET Contract NRC-CR-97-0002 and State of Texas Advanced Technology Program #0904.

REFERENCES

[1] McCulloch, W. S., and W. Pitts, A logical calculus of the ideas immanent in nervous activity, Bulletin of Mathematical Biophysics, 1943.

[2] Hebb. D. O. The Organization of Behavior: A Neuropsychological Theory, New York: Wiley 1949.

[3] Rochester, N., J. H. Holland, L. H. Habit, and W. L. Duda, Tests on a cell assembly theory of the action of the brain, using a large digital computer, IRE Transactions on information Theory, vol. IT-2, pp. 80-93.

[4] Minsky, M.L., Steps towards artificial intelligence, Proceedings of the Institute of Radio Engineers, vol. 49, pp. 8-30, 1961.

[5] Rosenblatt, F., The Perceptron: A probabilistic model for information storage and organization in the brain, Psychological Review, vol. 65, pp. 386-408, 1958

[6] Minsky, M.L. and S. A. Papert, Perceptrons, MIT Press Cambridge, MA 1969.

[7] Hopfield, J. J., Neural networks and physical systems with emergent collective computational abilities, Proceeding of the National Academy of Sciences, USA, vol. 79, pp. 2554-2558, 1982.

[8] Kohonen, T., Self-organized formation of topologically correct feature maps, Biological Cybernetics, vol. 43, pp. 59-69, 1982.

[9] Ackley, D. H., G. E. Hinton, and T. J. Sejnowski, A learning Algorithm for Boltzmann Machines, cognitive science, vol. 9, pp. 147-169, 1985.

[10] Rumelhart, D.E., G. E. Hinton, and R. J. Williams, Learning representations of back-propagation errors, Nature (London), vol. 323, pp. 533-536, 1986.

[11] Tom M. Mitchell, Machine Learning, McGraw-Hill Companies, Inc. 1997.

[12] Simon Haykin, Neural Networks a comprehensive foundation. Prentice-Hall, Inc. New Jersey 1999.

[13] G. F. Carey, A. Plaza, Local refinement of simplicial grids based on the skeleton. Applied Numerical Math., 32(2000), pp. 195-218.

[14] G.F.Carey, Computational Grids: Generation, Refinement, and Solution Strategies, Taylor and Francis, 1997.

[15] J.H.C Whitehead, On C1-Complexes, Ann. Math 41(1940) 809-824.

[16] Abani K. Patra, Atanas Pehlivanov, David Littlefield, Graham Carey and J. Tinsley, Error and Shape Impact Solution.

[17] M. Stynes, On faster convergence of the bisection method for all triangles, Math. Comp. 35 (1980) 1995-

1201. Quality Indicators for Adaptive Refinement of Deforming Finite Elements with Application to Lagrangian.
[18] M. C. Rivara, C. Levin, A 3D refinement algorithm suitable for adaptive and multi-grid techniques, J. Computational. Appl. Math. 8 (1992) 281-290.
[19] P. Conti, N. Hitschfeld, W. Fichtner, Ω-an octree-based mixed element grid allocator for simulation of complex 3D device structures, IEEE Trans. Comput. Aided design 10 (1991) 1231-1241.
[20] A. Liu, B. Joe, Relationship between tetrahedron shape measures, BIT 34 (1994) 268-287.
[21] A. Liu, B. Joe. On the shape of tetrahedra from bisection, Math. Comp. 63 (1994) 141-154.
[22] Linsker, R., Towards an organizing principle for a layered perceptual network, in Neural Information Processing Systems, D.Z. Anderson, ed., pp.485-494, New york: American Institute of Physics, 1988.
[23] Bell, A.J., and T. J. Sejnowski, An informative-maximization approach to blind separation and blind deconvolution, Neural computation, vol. 6, pp. 1129-1159, 1995.
[24] Broomhead, D.S., and D. Lowe, Multivariable functional interpolation and adaptive networks, complex systems, vol. 2, pp. 321-355, 1988.
[25] Vapnik, V. N., The Nature of statistical learning theory, New York: Springer-Verlag 1995.
[26] Vapnik, V. N., Principles of risk minimization for learning theory, Advances in Neural Information Processing Systems, vol. 4, pp. 831-838, San Mateo, CA: Morgan Kaufmann, 1992.
[27] Freeman, W. J., Societies of Brains, Hillsdale, NJ: Lawrence Erlbaum 1995.
[28] Knupp, P. Achieving Finite Element Mesh quality via optimization of the Jacobian Matrix and associated quantities, in press, 2000.
[29] Mead, C. A., Analog VLSI and Neural Systems, Reading, MA, Addison-Wesley 1989.
[30] Aspray, W., A. Burks, Papers of John von Neumann on Computing and Computer Theory, Charles Babbage Institute Reprint Series for the History of Computing, vol. 12. Cambridge, MA: MIT Press, 1986.